EP0498511A1

Publication Title:

Method of and apparatus for coding digital image data.

Abstract:

In the coding of image data corresponding to a raster of pixel values a generalized run length coding is used in respect of the difference between the pixel values in the raster line to be coded and the pixel values in the same position in the preceding raster line.

In multi-value pixels (grey values), the generalized run length coding comprises an approximation of the course of the differences as a function of the position in the raster line with a linear function. The code then contains the number of pixels and a characterisation of the approximating function (90).

In binary pixels (on/off) the length is simply determined of the series of pixel values each equal to the pixel value in the same position in the preceding raster line.

In a particularly effective embodiment, this generalized runlength coding with respect to th
32b
e preceding line is combined with generalized run length coding of the pixel values within a line. In this case both these methods are each time performed in parallel, and the method delivering the most effective code (that is the longest series of pixels, compensated for the length of the code itself) is selected for the actual encoding.

----

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 498 511 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **92200344.7**

(22) Date of filing: **07.02.92**

(51) Int. Cl.5: **H04N 1/41**, H04N 7/137,
H03M 7/30

(30) Priority: **08.02.91 NL 9100225**

(43) Date of publication of application:
12.08.92 Bulletin 92/33

(84) Designated Contracting States:
DE FR GB IT NL SE

(71) Applicant: **Océ-Nederland B.V.**
St. Urbanusweg 43
NL-5914 CC Venlo(NL)

(72) Inventor: **Moolenaar, Abraham**
Clematisstraat 30
NL-5925 BE Venlo(NL)

(74) Representative: **Hanneman, Henri W.A.M. et al**
Océ-Nederland B.V. Patents and Information
Postbus 101
NL-5900 MA Venlo(NL)

(54) Method of and apparatus for coding digital image data.

(57) In the coding of image data corresponding to a raster of pixel values a generalized run length coding is used in respect of the difference between the pixel values in the raster line to be coded and the pixel values in the same position in the preceding raster line.

In multi-value pixels (grey values), the generalized run length coding comprises an approximation of the course of the differences as a function of the position in the raster line with a linear function. The code then contains the number of pixels and a characterisation of the approximating function (90).

In binary pixels (on/off) the length is simply determined of the series of pixel values each equal to the pixel value in the same position in the preceding raster line.

In a particularly effective embodiment, this generalized runlength coding with respect to the preceding line is combined with generalized run length coding of the pixel values within a line. In this case both these methods are each time performed in parallel, and the method delivering the most effective code (that is the longest series of pixels, compensated for the length of the code itself) is selected for the actual encoding.

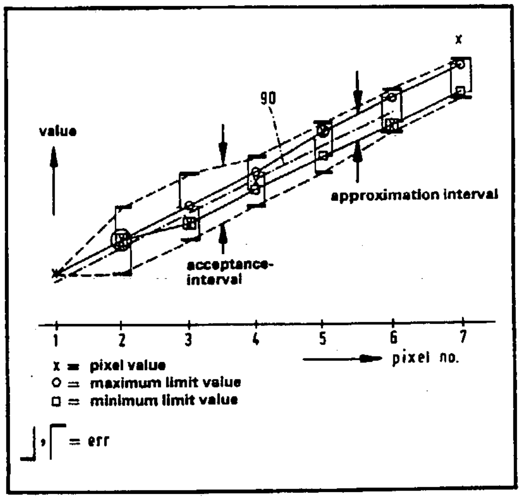

FIG. 9

Rank Xerox (UK) Business Services

The invention relates to a method of coding digital image data comprising consecutive blocks of data groups having as content the values of pixels of an image, comprising replacing series of contiguous data groups within a block that share a specific property by a code which contains the number of data groups in the series. The invention also relates to an apparatus for coding digital image data according to this method.

Digital image data form the digital representation of a physical image and is e.g. generated by electro-optically scanning the image according to an orthogonal raster of rows and columns of pixels and converting the measured optical density of each pixel into a digital value, that may have one of two possible values (black or white) or one of several possible values (shades of gray).

Normally, the digital data have the form of groups of bits or bytes and are stored in a memory in blocks. One such block may correspond to one row or line of pixels of the image.

A method of kind described above is known from IBM Technical Disclosure Bulletin, Vol.31, No. 5 (October 1988). In accordance with this known method, series of data groups (a data group being a byte with the grey value of a pixel) having an identical value within a line are coded with a code which contains (a) the number of data groups in the series and (b) the content of those data groups. This coding is generally termed "run length coding". Run length coding is also frequently applied to data groups containing binary pixel values, as is apparent, for example, from USP 4 748 512.

If, as in a page of text, the rows of pixels contain many series of identical value, this coding can result in a considerable compression of the image data, and this is an attractive feature in connection with storage thereof in a memory or with its transmission, for example, over a network.

However, there are many series of pixels with identical value in the column direction as well, and they can be used for a more extensive coding, which results in even greater compression of the image data.

The method according to the invention aims to improve the known method, and does attain this by comparing the content of each data group with the content of the data group in the same position in the preceding block and delivering a difference signal based on that comparison and replacing a series of contiguous data groups of which the said difference signal is by approximation a linear function of the position in the series by the said code.

Although the above U.S. patent also makes use of a vertical repetition code, it does so only to a very restricted degree, namely by replacing lines being completely identical to the preceding one by a repetition code. This is a relatively ineffective coding method, since there is relatively little chance that two consecutive lines will be completely identical.

When a uniform grey surface in an image is detected by a scanner, there will still be small variations in the measured pixel values due to noise and deviations in the scanner. These small variations restrict the codability if complete equality of the corresponding data groups is required. It has been found, however, that minor variations in grey values frequently do not visibly change the total picture, and consequently the choice has been made to make a series of consecutive data groups codable in the case of grey value pixels with minor adjustments of the pixel values. The method according to the invention therefore provides for equality "by approximation" and not absolute equality. As a result, in practice the coding results in a much higher compression.

Finally, the coding according to the invention is made even more effective by extending the equality criterion from the known method to the requirement that the contents of the data groups in a coded series should be a linear function of the position in that series. This requirement includes the following criteria:

- the difference between the corresponding data groups increases or decreases (linearly by approximation); this will for instance occur with a grey value changing in the diagonal direction;
- the corresponding data groups always differ (by approximation) an equal amount; this will occur in a grey value changing in the vertical direction.

A particular case of the second criterion is a difference equal to zero; this case can also be applied to binary pixel values, in which exact equality is required. In such cases the data groups may contain one or a number of (e.g. eight) pixel values. This particular embodiment provides a very powerful compression technique for binary image data. A data processing operation is described for both criteria in the claims and in the description.

It should also be noted that the approximation of a series of pixel values by a linear function is known per se and is described, for example, in IBM Technical Disclosure Bulletin, Vol.30, No. 12 (May 1988). However, the use of the pixel values in the preceding line to increase the efficiency of the coding cannot be derived therefrom.

In a further embodiment of the method according to the invention the above-described coding with reference to the preceding block is combined with run length coding within one block. This embodiment is characterised in that starting from a starting position in a block to be coded, and always for a following position within the block, the dif-

ference is determined between the content of the data group in that position and the content of the data group in the same position in the preceding block and a check is made whether the course of the differences thus determined is by approximation a linear function of the position in the block over all positions from the starting position and a replacement code for the longest series thus found is determined, in that starting from the same starting position in the block to be coded, and always for a following position within the block, a check is made whether the course of the contents of the data groups is by approximation a linear function of the position in the block over all positions from the starting position and a replacement code for the longest series thus found is determined,
and in that the one of both said longest series that delivers the most efficient code is selected and replaced by the associated replacement code.

Since the coding method according to the invention always has recourse to the data groups in the block preceding the block to be coded, decoding must also always make use of the preceding block. In decoding, however, this preceding block is the result of decoding of the code for that block. If this result is not exactly identical to the original block, and that is undoubtedly the case with the approximating coding as described above, then the decoded block will contain errors and increasingly greater deviations may occur due to decoding errors stacking up in consecutive blocks. In the method according to the invention, this problem is solved in that after coding of a series, the original content of the data groups of that series is replaced by a content corresponding to the result of coding and subsequent decoding of the data groups of that series.

In this way the decoding errors will never be greater than the inaccuracies in the code within a block.

The invention also relates to apparatus for performing the method according to the invention, which is characterised by means for comparing the content of each data group with the content of the data group in the same position in the preceding block and delivering a difference signal based on that comparison and means for finding series of contiguous data groups of which the said difference signal is by approximation a linear function of the position in the series and designating such series for coding. In these circumstances the linear function, in accordance with the selected form of the method, may in turn be a sloping linear function or a constant function, possibly with the value zero.

The invention also relates to apparatus performing the above-described combination of coding with reference to the preceding block and run length coding within one block.

The invention will now be explained by reference to the accompanying drawings wherein:

Fig. 1 is a flow diagram of the general form of the method according to the invention.

Fig. 2 is a flow diagram of a procedure for searching for a series which can be coded by run length coding.

Fig. 3 is a flow diagram of a procedure for searching for a series which can be coded by coincidence coding.

Fig. 4 is a flow diagram of a procedure for selecting and compiling a run length code or a coincidence code and writing this code into the file for the coded image.

Fig. 5 is a flow diagram of a procedure for compiling a copy code and writing this code into the file for the coded image.

Fig. 6A together with Fig. 6B and Fig. 6C is a flow diagram of a procedure for searching for a series which can be coded with run length coding.

Fig. 7 is a flow diagram of a procedure for searching for a series which can be coded with coincidence coding.

Fig. 8 is a flow diagram of a procedure for selecting and compiling a run length code or a coincidence code and writing said code into the file for the coded image.

Fig. 9 is a graphic showing a number of data groups in explanation of an approximation procedure according to the invention.

Fig. 10 is a block diagram of an apparatus according to the invention.

In the following description of the invention it has always been assumed that the data to be coded are supplied or stored in a memory in the form of bytes which correspond to the pixels of an image and which - just like those pixels - are arranged in consecutive lines. The bytes contain 8 bits which, in the case of binary pixels, correspond to the values of 8 adjacent pixels and, in the case of multi - value pixels (i.e. pixels which contain a grey value), correspond to one pixel value.

For the coding, a search is made in each line for series of bytes which can be described by means of a single code. Three coding methods are available for this, namely the run length coding (combining a series of identical bytes in a line), coincidence coding (referring to bytes situated at the same place in the preceding line) and copy coding (accepting the bytes to be coded into the code).

The series of bytes concerned in this are coded by a single code consisting of two or more bytes, which shows:

- the number of bytes in the series
- the method of coding
- the data further required for decoding.

The number of bytes and the method of coding are combined in the first code byte by dividing the values that this byte can assume into three intervals each corresponding to a coding method and increasing the number of bytes by a fixed amount so that their value appears in the associated interval. The 256 values which can be represented with 8 bits are divided, for example, into the intervals 1-100, 101-200 and 201-256, respectively corresponding to copy coding, run length coding and coincidence coding. The number of bytes is then increased, for example, by the lowest value of the applicable interval minus 1. Henceforth this amount is referred to as "OFF(CP)", "OFF(RL)" and "OFF-(CO)" respectively. The first code bytes for a series of 22 bytes coded in respect of run length will then contain the value 122, because OFF(RL) is in this example equal to 100.

The data further required for decoding vary with the method of coding and will be discussed in detail in the description thereof.

Fig. 1 is a flow diagram of the general form of the method according to the invention. It is based on an image described in a data file formed by H lines of L bytes. In each case k denotes the number of lines still to be coded and j the number of bytes still to be coded within the line. Three series counters RL, CO and CP are used which at each time contain the length of the codable series of bytes known at that time. RL contains the length of the run length series, CO that of the coincidence series and CP that of the series formed for copy coding.

At the start of the procedure, counter CP is first set to zero (101) and k is given the value H of the total number of lines of bytes (102). All the lines of bytes are then processed in a loop operation.

Whenever coding of a new line of bytes (103) is started, a test is made whether the end of the image has been reached (104), in which latter case coding is terminated. Otherwise j is given the initial value L (105) and then all the bytes in the line are successively processed by reducing j continually by 1(106). During the coding of a line a test is always carried out to check whether the end of the line has been reached (107) and if this really is the case any saved bytes for coding by copy coding are coded and written away in the file for the coded image (108), after which a new line is started (103).

At the start of a new line a check is made as to whether the first byte forms part of a series which can be coded by run length coding and, if so, how many bytes this series contains (109). A check is then made as to whether the first byte forms part of a series which can be coded with coincidence coding and, if so, how many bytes that series contains (110). The procedure during these checks is explained hereinafter with reference to Figs. 2 and 3, which relate to the case of binary pixels, and Figs. 6 and 7 which relate to the case of multi-value pixels.

The series lengths found are compared with a minimum value (hereinafter referred to as MIN(RL) and MIN(CO) for run length and coincidence coding respectively) to prevent very short series from being coded with run length or coincidence coding (111, 112). The reason for this is that copy coding of very short series requires less bytes than one of the other two coding methods, particularly when a number of very short series follow one another; in that case they can be combined in a copy code. It should be noted that the minimum values for run length and coincidence coding are different because the lengths of these codes themselves differ.

If at least one of the series satisfies this minimum requirement, then first of all any saved bytes for coding with copy coding are coded and written away in the file for the coded image (113). The most efficient coding method (run length or coincidence) is then determined, after which the bytes are coded and written away to the file for the coded image (114).

If the series do not satisfy either of their length criteria, copy coding is selected for the first byte. Since it is not yet known whether the next byte may have to be coded with copy coding as well, the first byte is not yet coded and written away, but is saved (this is done by increasing the counter CP by 1(115)). Only when a series has been found which can be coded with run length or coincidence coding, or when the end of the line is reached, the saved bytes are coded with copy coding and written away in the file for the coded image.

The entire procedure described is then repeated starting with the first still uncoded byte of the current line (106).

Fig. 2 is a flow diagram of a procedure for searching for a series which can be coded by run length coding, in the case of binary pixels, in which the bytes therefore each have the values of 8 pixels. This procedure forms a filling in of the block 109 in Fig. 1.

This procedure makes use of a counter "pointer" for continually indicating the byte to be processed. The procedure starts with initialization of the "pointer" at the position of the first still uncoded byte (201). The series counter RL is then set to 1(202). The following bytes are then successively processed in a loop operation. The loop operation starts with a check whether the maximum number of bytes to be coded in a code has already been reached (203), and a check as to whether the line end has been reached (204). If this is the case, the test is terminated and a return is made to the main line of the method in Fig. 1. Otherwise the value of the next byte (205) is compared with that of the byte on the starting position (206). If these values are identical, the counter RL is increased by 1(207), whereupon the next byte is checked. If the values are not identical, the test is terminated and a return is made to the main line of the method in Fig. 1. Thus RL always contains the number of bytes found to be coded with run length coding.

Fig. 3 is a flow diagram, again for the case of binary pixels, of a procedure for searching for a series which can be coded with coincidence coding. This procedure forms a filling in of the block 110 in Fig. 1.

This procedure makes use of a counter "pointer" for continually indicating the byte to be processed.

The procedure starts with initialization of the series counter CO on value 0 (301), followed by a test (302) as to whether there is a preceding line for comparing the bytes with. If not (i.e. the line under test is the first of the file) the procedure is not started, but a return is made to the main line of the method in Fig. 1. The "pointer" is then initialized at the position of the first still uncoded byte (303).

The following bytes are then successively processed in a loop operation. The loop operation starts with a test as to whether the maximum number of bytes to be coded in a code has already been reached (304), and a test as to whether the line end has been reached (305). If that is the case, the check is terminated and a return is made to the main line of the method in Fig. 1.

Otherwise, the value of the next byte (306) is compared with that of the byte in the same position in the preceding line (307). If these values are identical, the counter CO is increased by 1(308), whereupon the next byte can be checked. If the values are not identical, the test is terminated and a return is made to the main line of the method in Fig. 1. CO thus always contains the number of bytes found to be coded with coincidence coding.

Fig. 4 is a flow diagram, again for the case of binary pixels, of a procedure for selecting and compiling a run length or a coincidence code and writing this code into the file for the coded image. This procedure, which forms a filling in of the block 114 in Fig. 1, is reached after it has been established that at least one codable series of sufficient length has been found.

A check is first made as to which method of coding is the most efficient, i.e. yields the fewest code bytes in relation to the number of original bytes. In this example, this is implemented as the coding method with which the most byte scan be coded and hence the length of the various codes themselves is disregarded. An extension of this kind, however, is obvious. The values of RL and CO are compared (401) for the said selection. A test is also made as to whether the series of the coding method selected satisfies its minimum length requirement (402,403). If this is not the case, then the other method of coding is finally selected.

If the run length coding is selected, the code is compiled from two sub-codes each defining a byte. The first sub-code, "code(A)RL", contains the number of bytes of the series less the minimum number MIN(RL) (in order to utilise to the maximum the value interval available for this coding method) and plus the fixed amount OFF(RL) already referred to, which is intended to cause the code to appear in the interval of the run length coding (404). The second sub-code, "code(B)RL", contains the value of the bytes in the coded series (405). The compiled code is then added to the file for the coded image (406). After the pointer j has been set to the last coded byte (407) a return is again made to the main line of the method in Fig. 1.

For decoding, the series is reconstructed by repeating the second byte a number of times, which arises out of the first byte.

If coincidence coding is selected, the code is formed by a byte "codeCO", which contains (408) the number of bytes of the series less the minimum number MIN(CO) and plus the fixed amount OFF(CO) already referred to, which is intended to cause the code to appear in the coincidence coding interval. This code is then added to the file for the coded image (409). After the pointer j has been set to the last coded byte (410), a return is made to the main line of the method in Fig. 1.

For decoding, the series is reconstructed by accepting from the previous line the corresponding byte for a number of times which arises out of the code byte.

Fig. 5 is a flow diagram of a procedure for compiling a copy code and writing this code into the file for the coded image. This procedure forms a filling in of the blocks 108 and 113 in Fig. 1.

Because of the way in which the bytes to be coded in this way are collected, it is not impossible that this procedure may be called up while there are no bytes to be coded. The procedure is therefore started with a test for this (501). If there really are no bytes to be coded, then there is immediately a return to the main line of the method in Fig. 1.

If the number of saved bytes is larger than the maximum number MAX(CP) to be reproduced in the code (test 502), the saved bytes are first coded in blocks of the maximum size and added to the file for the coded image until the number of saved bytes still to be coded has fallen to below the maximum. If the number of bytes to be coded is larger than MAX(CP), a block of MAX(CP) bytes is thus coded.

The code is compiled from two sub-codes, the first of which "code(A)CP" occupies a byte (503). This first sub-code contains the number of bytes of the series less the minimum number MIN(CP) (which in this example has the value 1) and plus the fixed amount OFF(CP) previously referred to, which is intended to cause the code to appear in the copy coding interval, and which in this example has the value 0.

The second sub-code "code(B)CP", contains the values of all the bytes in the coded series and therefore has a length equal to the number thereof, MAX(CP) (504). After compilation of the code, it is written into the file for the coded image (505) and the counter containing the number of bytes to be coded with copy coding is reduced by the value MAX(CP) (506). The procedure is then repeated with the rest of the bytes to be coded.

If the number CP of bytes to be coded is less than the maximum number MAX(CP), a similar procedure is carried out with the smaller number of bytes (507, 508, 509) and after counter CP has been set to zero (510) a return is made to the main line of the method in Fig. 1.

For decoding, the series is reconstructed by reproducing the bytes following the first code byte. The number thereof follows from the first code byte.

The above method is very suitable for coding an image with binary pixels, for when pixels can have only two values there is a relatively considerable chance of two pixels having the same value. The method is additionally also usable in principle for coding multi-value pixels (pixels having a grey value). The conventional practical technique will be taken as the basis hereinafter, in which the value of a pixel is reproduced by a byte (8 bits). It will also be assumed that the bytes concerned are arranged in lines corresponding to the physical arrangement of the pixels in the image.

If the criterion of equality of pixel values, as used hereinbefore, is taken over arbitrarily, the method is relatively inefficient. In the above-described case of one byte per pixel there are 256 grey values possible and hence there is little chance of two pixels having exactly the same value. Minor differences in pixel values can also readily occur in a uniform grey surface, for example due to noise. Thus coding based on equality of bytes would in practice yield little compression.

Since it has been found that very small differences in grey values are barely visible, if at all, in a print (the human eye is much more sensitive to considerable jumps in grey values than to minor ones), there is no necessity for complete equality. It is sufficient to require equality within a certain margin. In this way it becomes possible, in the case of run length coding, to code the values of a series of consecutive pixels by means of a geometrical function which does not demand complete equality with this function but equality within a certain margin.

On the other hand, in the case of multi-value pixels, it is possible to implement the concept of coincident coding in the form of a coding of the difference between the values of corresponding pixels in the current line and the preceding line. In this way coincidence coding for grey values changes to run length coding of the said difference.

The invention provides an approximation for consecutive pixel values (each represented by a byte) with a linear function. Two variants thereof will be described hereinafter. In the first variant, the direction of the linear function can be adapted to the values to be approximated, while in the second variant the linear function is a constant function. Both variants can be used for coding both a run length of pixel values in a line and a run length of the difference between corresponding bytes in the current line and the preceding line (generalized coincidence coding), but in the following example each is used for one of the two.

Starting from a first byte, the two variants form an acceptance interval within which the next byte must be situated if the approximation is to be successful, and an approximation interval. The acceptance interval converges with the processing of an increasing number of bytes, to a predetermined width "err" and is always so placed that all the processed bytes are contained. The approximation interval diverges at the same time to the same width "err" and finally acquires the same position as the acceptance interval.

In the first variant, the direction is initially determined by placing the linear function through the first two bytes to be coded, but as soon as a subsequent byte is found no longer to be situated within the acceptance interval, the approximation is repeated with a new direction. This new direction is found by correcting the old direction with the difference in value of the byte no longer situated within the acceptance interval and the closest boundary of the acceptance interval divided by the number of bytes by which the byte no longer situated within the acceptance interval is spaced from the starting byte. If this change of direction does not cause any of the bytes already processed to be outside the acceptance interval, the number of bytes approximatable by this new direction has increased by 1, because the last byte will now also be situated within the acceptance interval. Otherwise, the number of approximatable bytes will become smaller and the new approximation is therefore less successful than the preceding one. In that case the preceding approximation is used for the coding.

The linear function having the found direction and passing through the centre of the approximation interval of the last approximated byte is now selected as approximation.

Fig. 9 gives an example of the effect of the first variant. For a number of consecutive pixels indicated by a serial number 1 to 7, the value is graphically represented by an x. The maximum and minimum limit values are always indicated by a small circle and a small square respectively. They define the approximation interval of which the change from one pixel to another is drawn in by a solid line. The acceptance interval, which is defined by, on the one hand, the maximum limit value minus a predetermined margin err, and, on the other hand, the minimum limit value plus the same margin err, is defined by two horizontal lines; their course for consecutive pixels is shown by broken lines. The direction of the connection of the value of the first pixel and that of the second pixel is taken as the approximation direction.

The acceptance interval of a specific pixel is found by moving the acceptance interval of the previous pixel along the approximation direction. If the value of the next pixel is within its acceptance interval but outside the approximation interval moved in the approximation direction (i.e.: above the maximum limit value or below the minimum limit value), then the associated limit value is made equal to the value of that pixel, so that the acceptance interval is also changed, because it is coupled to the limit values. In the example of Fig. 9, the value of 7th pixel is no longer within its acceptance interval so that the approximation applies to the first 6 pixels. The linear approximation function is now formed by the straight line passing through the centre of the approximation interval of the 6th pixel and having the approximation direction as direction. In the drawing this line is denoted by a dash-dot line (90).

A check is now made as to whether more pixels can be described with another approximation direction. This is no longer shown in Fig. 9 since it would otherwise become unclear. The new approximation direction is calculated by dividing by 6 the difference between the value of the 7th pixel and the top limit of its acceptance interval, adding the result of this to the value of the 2nd pixel and drawing the connecting line between the point thus found and the first pixel. The new approximation direction is the direction of this connecting line. The entire approximation procedure is then repeated with this new approximation direction. If it is possible to describe more pixels in this way, a new approximation direction is again determined and the approximation procedure repeated. If an equal number of or less pixels are described with a new approximation direction, the approximation is terminated and the last approximation found is taken as adequate.

The second variant of the approximation is a simplified form of the approximation according to the first variant. In this the approximation function selected is the constant function passing through the centre of the approximation interval of the last approximated byte. The procedure is largely identical to that of the first variant except that no approximation direction is calculated and there is therefore no iteration with different approximation directions.

Figs. 6A, 6B and 6C together show, for the case of multi-value pixels, where the bytes each have the value of 1 pixel, a flow diagram of a procedure for searching for a series which can be coded with run length coding in the extended meaning (approximation with a linear function, in this case in accordance with the said first variant of the approximation method). The diagram of Figs. 6A-6C forms the filling in of block 109 in Fig. 1.

After initialization of the counter RL (to value 1; step 601) and of the pointer (to the position of the first byte to be coded; step 602), a variable "offset" is provided of a value equal to the difference in value between the byte at the starting position and the next byte (603). This "offset" determines the direction of the linear approximation function. A variable "starting value" is then filled in with the value of the byte at the starting position (604). This "starting value" determines the position of the linear approximation function. The values of "offset" and "starting value" are provisional values with which a first approximation attempt is carried out. The results of this first approximation attempt, which are expressed in adjusted values for "offset" and "starting value", are again used for an approximation attempt as described above.

These successive approximation attempts are performed in a loop operation which now follows. The loop operation starts with an initialization of an internal counter RLX which indicates how many bytes with this approximation can be coded (605). Each approximation loop again contains an internal loop operation for counting the number of codable bytes. In this internal loop the byte to be processed is always indicated by a current variable "pointer 2". Before entering the loop, the "pointer 2" is first filled with the position of the first byte to be coded (606) and a maximum limit value "UL" and a minimum limit value "LL" are filled with the value of the byte at the starting position plus the value "offset". On the first approximation run this corresponds to the value of the first byte after the starting position itself. The maximum and minimum limit values are used to define an approximation interval which will later serve as a basis for the approximation function.

The internal loop operation starts with a test as to whether the maximum number of bytes to be coded with a code has been reached (608) and whether the line end has been reached (609), in which cases the approximation attempt is terminated and the procedure jumps to the evaluation step to be described hereinafter.

For the next byte (610) a check is made whether the value of this byte is situated within the acceptance interval, which acceptance interval is defined by "LL+err" and "UL-err", where "err" is a predetermined margin (611, 612). If this is the case, a counter RLX is increased by 1 (613), otherwise this approximation attempt is terminated and the procedure skips to the previously mentioned evaluation step.

If the value of the indicated byte is within the aceptance interval, then a check is made whether this value is also situated in the approximation interval defined by LL and UL (614, 616). If this is not the case, then the value of LL or UL is first shifted to the value of he byte (615, 617), so that this byte also comes to lie within the approximation interval. It will be clear that in this way the approximation interval and the acceptance interval finally approach one another and also approach a width of "err". If the value of the indicated byte is within the approximation interval, treatment of the next byte in the line is started after the values of "LL" and "UL" have first been adapted by increasing them by "offset" (618).

In the evaluation step a check is made as to whether it is appropriate to carry out a new approximation attempt with new initial values. For this purpose, the run length RLX just found is compared with a run length RL (619) found in a possible previous approximation. If RLX is found to be larger than RL or if there is no previous approximation (in the latter case RL still has the initialization value 1), a new attempt is made with adapted initial values. In the other cases it is assumed that the previous approximation was the best and the procedure skips back to the main line of the method in Fig. 1, taking with it the values of RL, "offset" and "starting value" from the previous approximation. "Starting value", "offset" and RL describe the approximation function completely.

If the completed approximation appears to be better than the previous one, the value of RL is made equal to the newly found value RLX (620) and the approximation function is pushed up so that it passes through the centre of the approximation interval of the last byte described by the approximation. This results in a new value for "starting value" (621).

Before calculating a new value for "offset" for a new approximation attempt, a check is now first made as to whether the end of the line or the maximum number of bytes to be coded with one code has been reached with the approximation just concluded (622). If this is the case, then it is pointless to make a better approximation attempt and a return is made to the main line of the method in Fig. 1 with the results of the last approximation.

In the other cases, the approximation function is rotated about the current starting value to an extent such that the first byte no longer accepted in the approximation attempt which has just concluded just comes within the associated acceptance interval. This is achieved by correcting the value of "offset" with the difference between the value of that byte and the top limit of its acceptance interval scaled to the distance between two consecutive pixels (623, 624, 625). A new approximation attempt is then carried out with the new values of RL, "starting value" and "offset".

Fig. 7 describes, for the case of multi-value pixels, searching for a series which can be coded with coincidence coding in the extended sense (approximation of the difference between the bytes and the corresponding bytes in the previous line with a linear function, in this case in accordance with the said second variant of the approximation method). The diagram of Fig. 7 forms a filling in of block 110 in Fig. 1.

This procedure starts with initialization of the counter "CO" to the value 0 (701), followed by a check as to whether the line of bytes to be coded is the first in the file (702). If this is the case, so that it is therefore impossible to compare the bytes of this line with the bytes of a previous line, then the procedure is immediately abandoned. A current variable "pointer" is then initialized at the position prior to the first byte to be coded (703). For the first byte to be approximated, a maximum limit value "UL" and a minimum limit value "LL" are determined by making both equal to the difference of the value of this byte and the value of the byte in the same position in the preceding line (704). The maximum and minimum limit values enclose an approximation interval which will later serve as a basis for the approximation function.

For the continuously following bytes a check is then made in a loop operation as to whether the difference between their value and the value of the corresponding byte in the preceding line can be approximated with a constant function. The loop operation starts with a check as to whether the maximum number of bytes to be coded with a code has not been reached (705) and whether the line end has not been reached (706), in which cases the approximation attempt is concluded.

The value of the next byte (707) is then compared with that of the byte in the same position in the preceding line (708). If the difference, "diff", is situated in the acceptance interval, which acceptance interval is defined by LL+err and UL-err (where err is a predetermined margin; 709,710), then the counter CO is increased by 1 (711), otherwise the approximation attempt is concluded.

If the value of the indicated byte is within the acceptance interval, then a check is made as to whether this value is also situated in the approximation interval defined by LL and UL (712, 714). If this is the case, the procedure passes to the next byte, otherwise the value of LL, or the value of UL respectively, is first shifted to the value of the byte, so that this byte also comes to lie within the approximation interval (713, 715). It will be clear that in this way the approximation interval and the acceptance interval finally approach one another and also approach a width equal to err.

On conclusion of the approximation attempt, the approximation value "displacement" is calculated by determining the average value of maximum and minimum limit values of the last byte described by the approximation (716). The procedure then returns to the main line of the method in Fig. 1. "Displacement" and CO describe the approximation function completely, the value of the bytes thus coded being found by adding the value "displacement" to the value of the corresponding bytes in the preceding line.

Fig. 8 describes - again for the case of multivalue pixels - the selection and compilation of a run length or a coincidence code and the writing of this code into the file for the coded image. This procedure is achieved after it has been established that at least one codable series of adequate length has been found. The diagram of Fig. 8 forms the filling in of the block 114 in Fig. 1.

A check is first made as to what coding method can be used for coding most bytes by comparing the values of RL and CO (801). As an alternative, it would also be possible to take into account here the length of the code itself, thus giving maximum efficiency.

In the event that the coding result of the run length coding from Figs. 6A - 6C is selected, the code is compiled from three sub-codes each occupying one byte. The first sub-code, "code(A)RL", contains the number of bytes of the series less the minimum number MIN(RL) (in order to provide maximum utilisation of the value interval available for this coding method) plus the previously mentioned fixed amount OFF(RL) which is intended to make the code appear in the interval of the run length coding (802).

The second sub-code, "code(B)RL", contains the starting value of the linear function found with which the bytes in the series are approximated (803). The third sub-code, "code(C)RL", contains the direction of the approximation function in the form of the value "offset" (804). The compiled code is then added to the file for the coded image "805".

For decoding, the series is reconstructed by calculating the value in accordance with the linear function characterised in the second and in the third byte, for a number of bytes following from the first byte.

The pointer j is then moved to the last coded byte (806) and the original values of the bytes in the memory are replaced by the values following from the coding, so that the latter values can serve as a reference for a possible coincidence coding of the bytes in the next line (807). In this way it is not possible for there to be any accumulation of errors as a result of the approximation in the coding.

In the event of the result of the coincidence coding from Fig. 7 being selected, the code is compiled from two sub-codes each occupying one byte. The first sub-code, "code(A)CO", contains the number of bytes of the series less the minimum number MIN(CO) plus the previously mentioned fixed amount OFF(CO) which is intended to cause the code to appear in the coincidence coding interval (808). The second sub-code, "code(B)-CO), contains the "displacement" value found during the approximation (809). The compiled code is then added to the file for the coded image (810).

For decoding, the series is reconstructed by taking over for a number of times following from the first code byte, the corresponding byte from the previous line plus the displacement value given in the second code byte.

The pointer j is then moved to the last coded byte (811) and again the original values of the bytes in the memory are replaced by values following from the coding (812).

Finally the procedure returns to the main line of the method in Fig. 1.

The compilation of a copy code and the writing of this code into the file for the coded image proceeds in exactly the same way as with binary bytes (see Fig. 5) and is therefore not described again.

Fig. 10 is a block diagram of apparatus according to the invention. This apparatus has a supply line 1 for the image data to be supplied to the apparatus, e.g. from a scanner or from a mass memory (not shown). This supply line is connected to a memory 2 for the storage of at least some of the image data, e.g. two lines of data groups, the term "data groups" denoting, for example, bytes which contain pixel values.

A pointer unit 3 is connected to the memory 2. A counter unit 4 is also connected to the memory 2. Counter unit 4 contains a comparator unit 5 and a first scout unit 6, interconnected and both connected to the connection between the counter unit 4 and the memory 2. The counter unit also contains a second scout unit 7 which is also connected to the connection between the counter unit 4 and the memory 2. Within the counter unit 4 the scout units 6 and 7 are connected to an assessment circuit 8. The counter unit 4 is connected to a code generator 9. A control unit 10 is connected to the pointer unit 3, counter unit 4 and code generator 9. Code generator 9 also has an output line 11 for outputting coded image data, for example, to a mass memory (not shown).

The units 3 to 10 are program modules in a computer, but could also be in the form of hardware circuits.

The apparatus operates as follows. Image data are input from outside to the memory 2 and stored therein at memory addresses corresponding to the position of the pixels whose value they contain in the image. For coding, there is no need to store more than two lines of data units: the line to be coded and the line preceding the same (if there is such a line).

Coding then starts. It is described below on the assumption that the coding has already advanced somewhat and that part of a line has already been coded. On the command of the control unit 10 the pointer unit 3 indicates the first still uncoded data group of the line to be coded, and then, again on the command of the control unit 10, the counter unit 4 checks whether the data groups following the position indicated by the pointer unit 3, together with the indicated data group, form a codable series, either a series codable by run length, in which case the operations in Fig. 2 or Figs. 6A-6C are carried out by the second scout unit 7, or a series codable by coincidence, in which case the operations in Fig. 3 or Fig. 7 are carried out by the first scout unit 6, which in so doing makes use of the differences between corresponding data groups in the two lines as calculated by the comparator unit 5.

The end results of the two scout units 6 and 7 are passed to the assessment circuit 8 which compares the lengths of the found series with the predetermined minimum lengths. If neither series satisfies this criterion, the assessment circuit 8 passes this on to the control unit 10, which increases by 1 a counter for the data groups to be coded by copy coding, and then orders the pointer unit 3 to indicate the next data group in the memory 2.

If at least one of the two series found satisfies its minimum length criterion, then the assessment circuit 8 passes the lengths found to the code generator 9 and reports this to the control unit 10. The latter then gives the code generator 9 the command for, first of all, converting to a code (in accordance with the procedure of Fig. 5), any data groups still waiting and intended for coding by copy coding and then for selecting the series which can be most efficiently coded and converting the same to a code (in accordance with the procedure of Fig. 4 or 8). The codes are output from the apparatus via the output line 11.

Whenever the code generator 9 has converted a series of data groups into a code, it generates a series of new data groups from the code and places it in the memory 2 at the position of the data groups just coded, to ensure that the correct reference is used for coding the next line of data groups. For decoding, of course, the original data groups are no longer available and thus it is only possible to use the decoded data groups.

When the code generator 9 has finished the coding, it reports this to the control unit 10, which then sets the pointer 3 to the first still uncoded data group, whereupon the entire procedure described above is repeated.

Further details of the operation of the apparatus will be immediately apparent from the description of the method and therefore will not be described in greater detail here.

The above description is only one example of an embodiment of the invention as contained in the claims. It will be clear to the skilled addressee that a number of other embodiments are possible within the scope of the claims.

Claims

1. A method of coding digital image data comprising consecutive blocks of data groups having as content the values of pixels of an image, comprising replacing series of contiguous data groups within a block that share a specific property by a code which contains the number of data groups in the series,
   characterised by
   comparing the content of each data group with the content of the data group in the same position in the preceding block and delivering a difference signal based on that comparison and
   replacing a series of contiguous data groups of which the said difference signal is by approximation a linear function of the position in the series by the said code.

2. A method of coding digital image data comprising consecutive blocks of data groups having as content the values of pixels of an image, comprising replacing series of contiguous data groups within a block that share a specific property by a code which contains the number of data groups in the series,
   characterised in that,
   starting from a starting position in a block to be

coded, and always for a following position within the block, the difference is determined between the content of the data group in that position and the content of the data group in the same position in the preceding block and a check is made whether the course of the differences thus determined is by approximation a linear function of the position in the block over all positions from the starting position and a replacement code for the longest series thus found is determined,

in that starting from the same starting position in the block to be coded, and always for a following position within the block a check is made whether the course of the contents of the data groups is by approximation a linear function of the position in the block over all positions from the starting position and a replacement code for the longest series thus found is determined, and

in that the one of both said longest series that delivers the most efficient code is selected and replaced by the associated replacement code.

3. A method according to claim 1 or 2, in which at least one said linear function is a constant function.

4. A method according to claim 1,2 or 3, wherein after a series of data groups has been coded the original content of the data groups of that series is replaced by a content corresponding to the result of coding and subsequent decoding of the data groups of that series.

5. A method according to claim 3, in which the said constant function has the value zero.

6. Apparatus for coding digital image data comprising consecutive blocks of data groups, having as content the values of pixels of an image, according to a method comprising replacing series of contiguous data groups within a block that share a specific property by a code which contains the number of data groups in the series,

characterised by

means (5) for comparing the content of a data group with the content of the data group in the same position in the preceding block and delivering a difference signal based on that comparison and

means (6) for finding series of contiguous data groups of which the said difference signal is by approximation a linear function of the position in the series and designating such series for coding.

7. Apparatus according to claim 6, comprising

- a memory (2) for storing at least part of the image data,
- a pointer (3) connected to the memory (2) for indicating a position in a block to be coded,
- a counter (4) connected to the memory (2) for counting, starting from the position indicated by the pointer (3), contiguous data groups within the block which together form a codable series, the counter (4) being provided with
  . a comparator (5) for comparing the content of a data group in the block to be coded with the content of the data group in the same position in the preceding block and delivering a difference signal based on that comparison and
  . a scout unit (6) connected to the comparator (5) and to the memory (2) for checking, starting from a position indicated by the pointer (3) and always for a following position within the same block, whether the course of the difference signals from the comparator (5) is by approximation a linear function over all positions from the starting position indicated by the pointer (3), and
- a code generator (9) connected to the counter (4) and to the memory (2) for delivering a code to replace data groups counted by the counter (4).

8. Apparatus according to claim 7, in which the counter (4) is designed for counting two series of data groups, both starting at the same starting position, and is therefore further provided with

  . a second scout unit (7) connected to the memory (2) for checking, starting from the position indicated by the pointer (3) and always for a following position within the same block whether the course of the contents of the data groups is by approximation a linear function overall positions from the starting position indicated by the pointer (3)

and in which the code generator (9) is provided with means for selecting, from the two series counted by the counter (4), the series that delivers the most efficient code.

9. Apparatus according to claim 6, 7 or 8,in which at least one of said linear functions is a constant function.

10. Apparatus according to claim 9, in which the said constant function has the value zero.

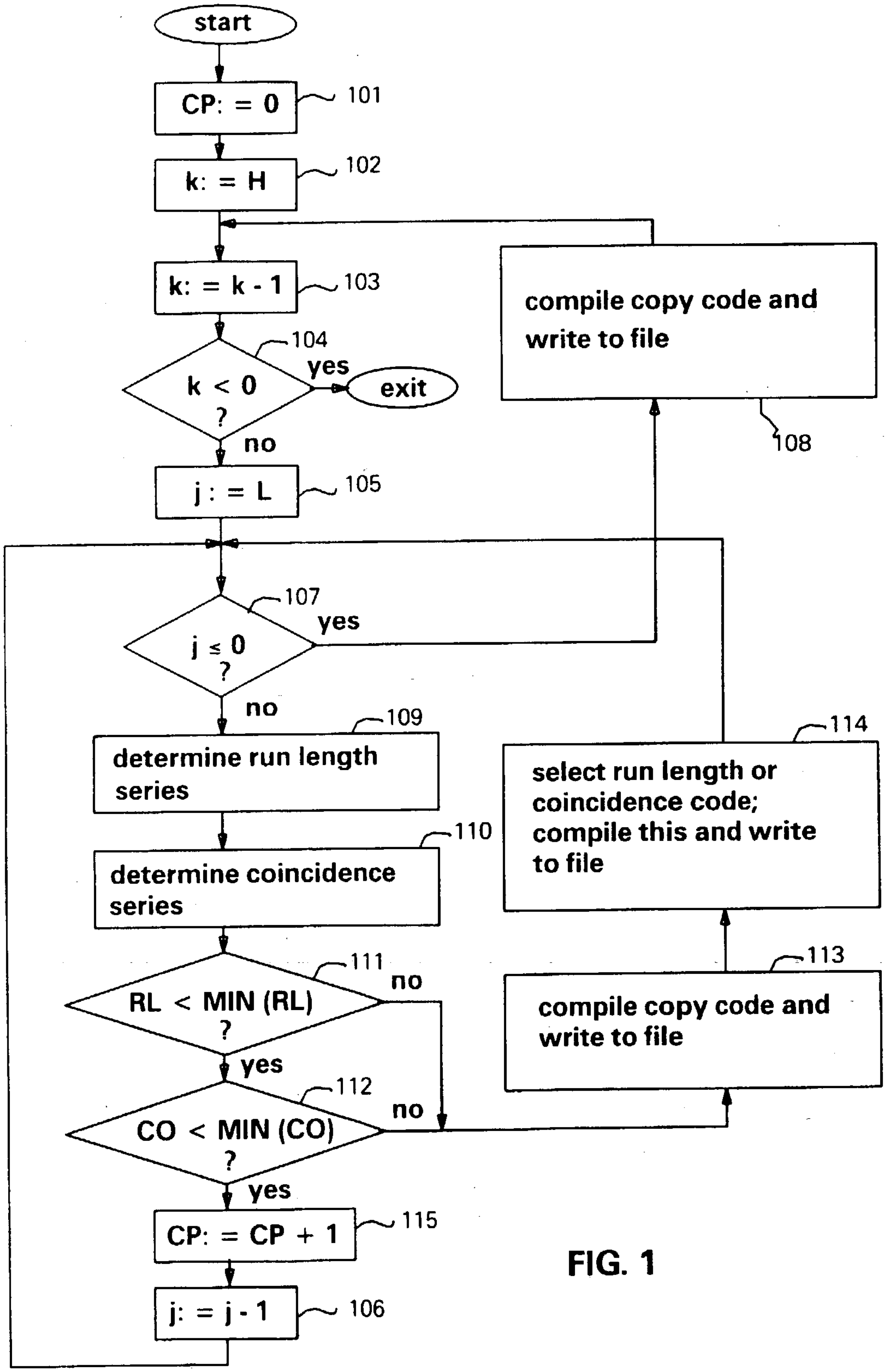
start
CP: = 0
101
k: = H
102
k: = k - 1
103
104
k < 0
?
yes
exit
no
j : = L
105
compile copy code and write to file
108
107
j ≤ 0
?
yes
no
109
determine run length series
110
determine coincidence series
111
RL < MIN (RL)
?
no
yes
112
CO < MIN (CO)
?
no
yes
CP: = CP + 1
115
j: = j - 1
106
114
select run length or coincidence code; compile this and write to file
113
compile copy code and write to file

FIG. 1

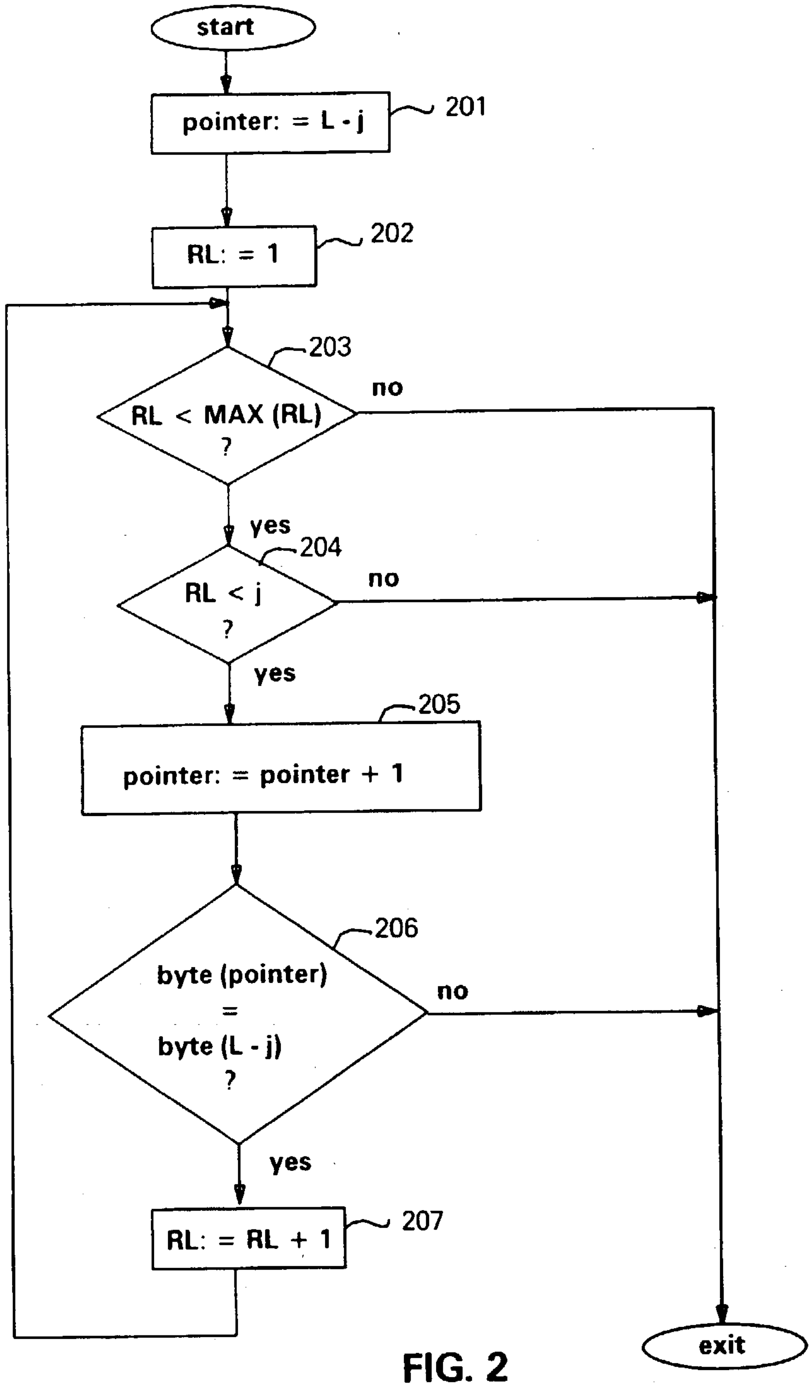
start
pointer: = L - j
201
RL: = 1
202
203
RL < MAX (RL)
?
no
yes
204
RL < j
?
no
yes
205
pointer: = pointer + 1
206
byte (pointer)
=
byte (L - j)
?
no
yes
207
RL: = RL + 1
exit

FIG. 2

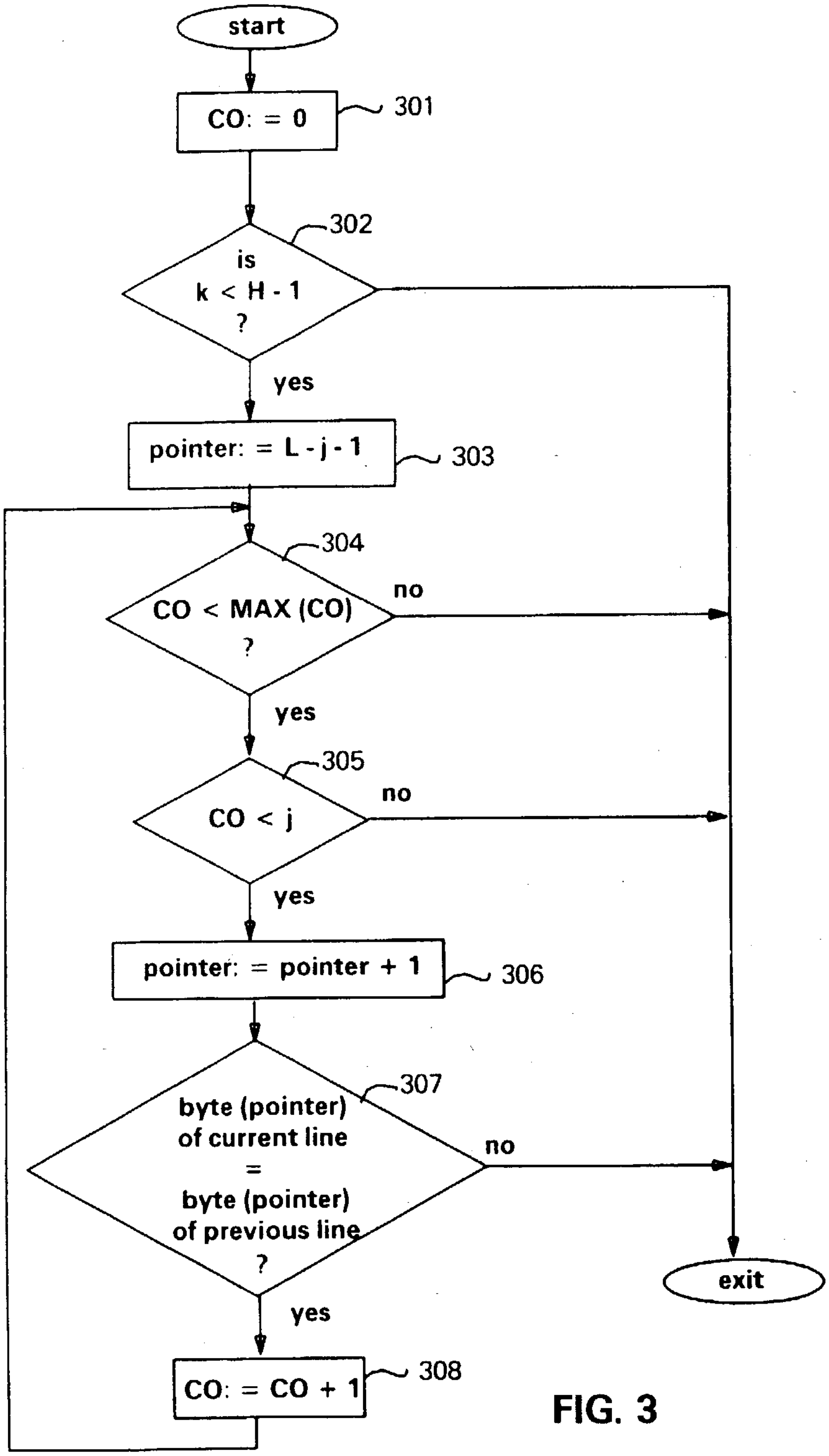
start
CO: = 0
301
302
is
k < H - 1
?
yes
pointer: = L - j - 1
303
304
CO < MAX (CO)
?
no
yes
305
CO < j
no
yes
pointer: = pointer + 1
306
307
byte (pointer)
of current line
=
byte (pointer)
of previous line
?
no
yes
CO: = CO + 1
308
exit

FIG. 3

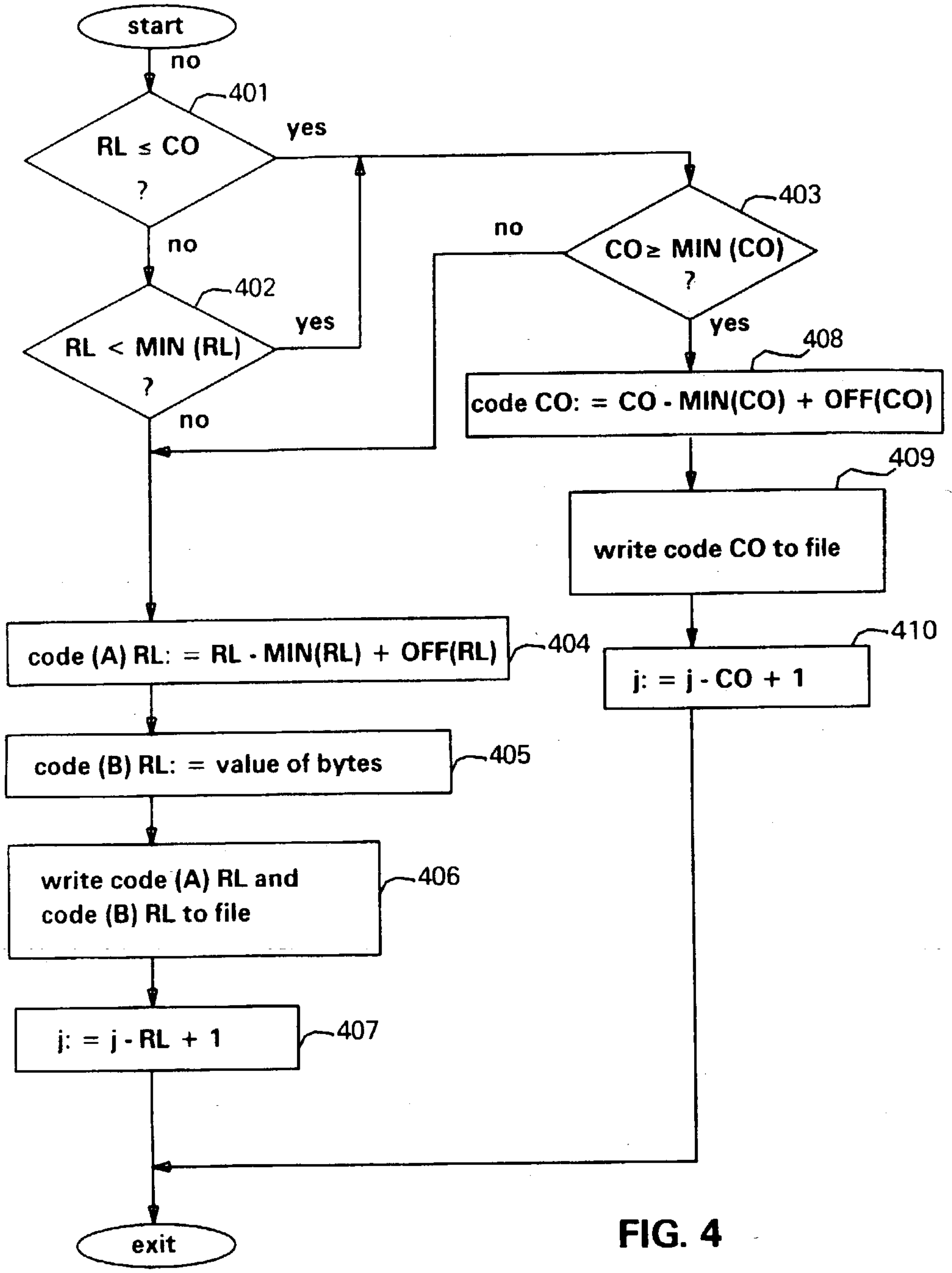
start
no
401
RL ≤ CO ?
yes
no
402
RL < MIN (RL) ?
yes
no
403
CO≥ MIN (CO) ?
no
yes
408
code CO: = CO - MIN(CO) + OFF(CO)
409
write code CO to file
410
j: = j - CO + 1
404
code (A) RL: = RL - MIN(RL) + OFF(RL)
405
code (B) RL: = value of bytes
406
write code (A) RL and code (B) RL to file
407
j: = j - RL + 1
exit

FIG. 4

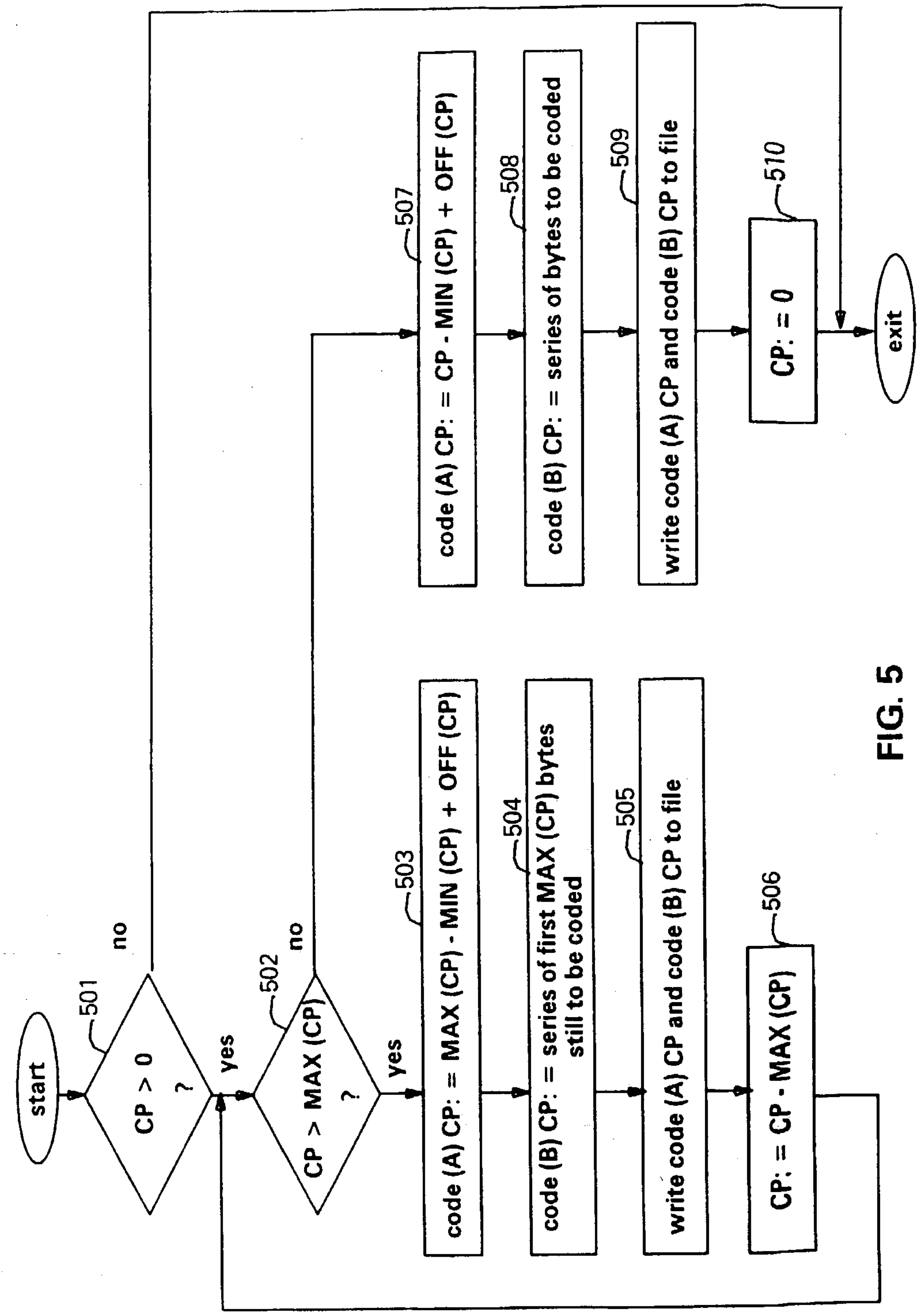
start
501
CP > 0
?
no
yes
502
CP > MAX (CP)
?
no
yes
503
code (A) CP: = MAX (CP) - MIN (CP) + OFF (CP)
504
code (B) CP: = series of first MAX (CP) bytes still to be coded
505
write code (A) CP and code (B) CP to file
506
CP: = CP - MAX (CP)
507
code (A) CP: = CP - MIN (CP) + OFF (CP)
508
code (B) CP: = series of bytes to be coded
509
write code (A) CP and code (B) CP to file
510
CP: = 0
exit

FIG. 5

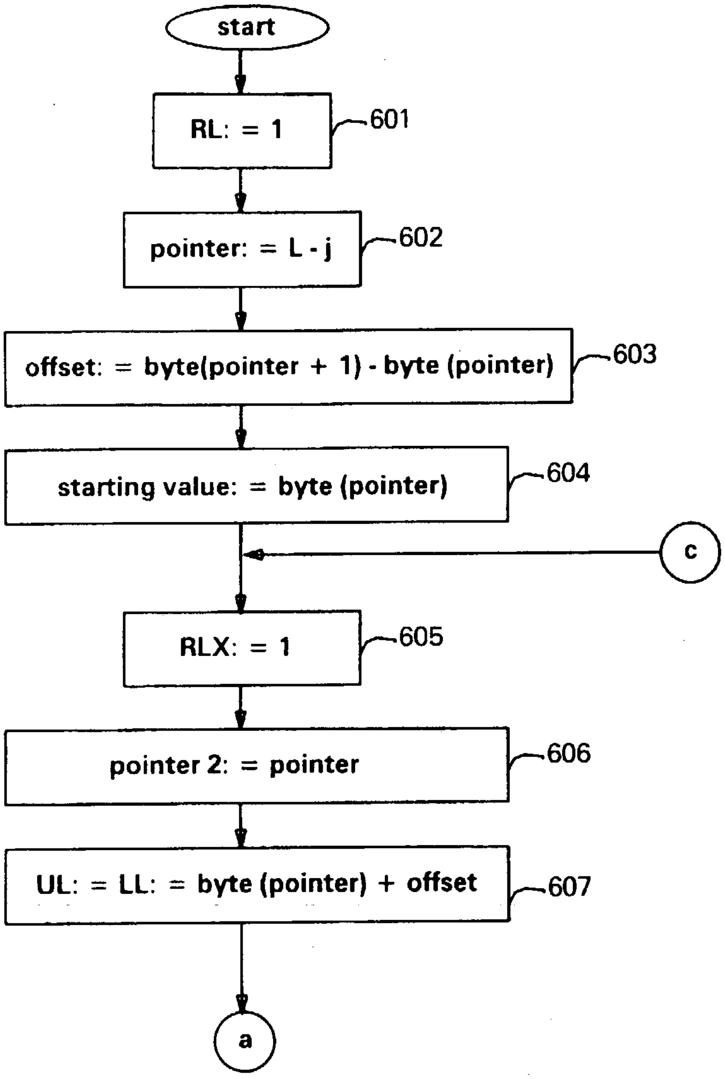
start
RL: = 1
601
pointer: = L - j
602
offset: = byte(pointer + 1) - byte (pointer)
603
starting value: = byte (pointer)
604
c
RLX: = 1
605
pointer 2: = pointer
606
UL: = LL: = byte (pointer) + offset
607
a

FIG. 6A

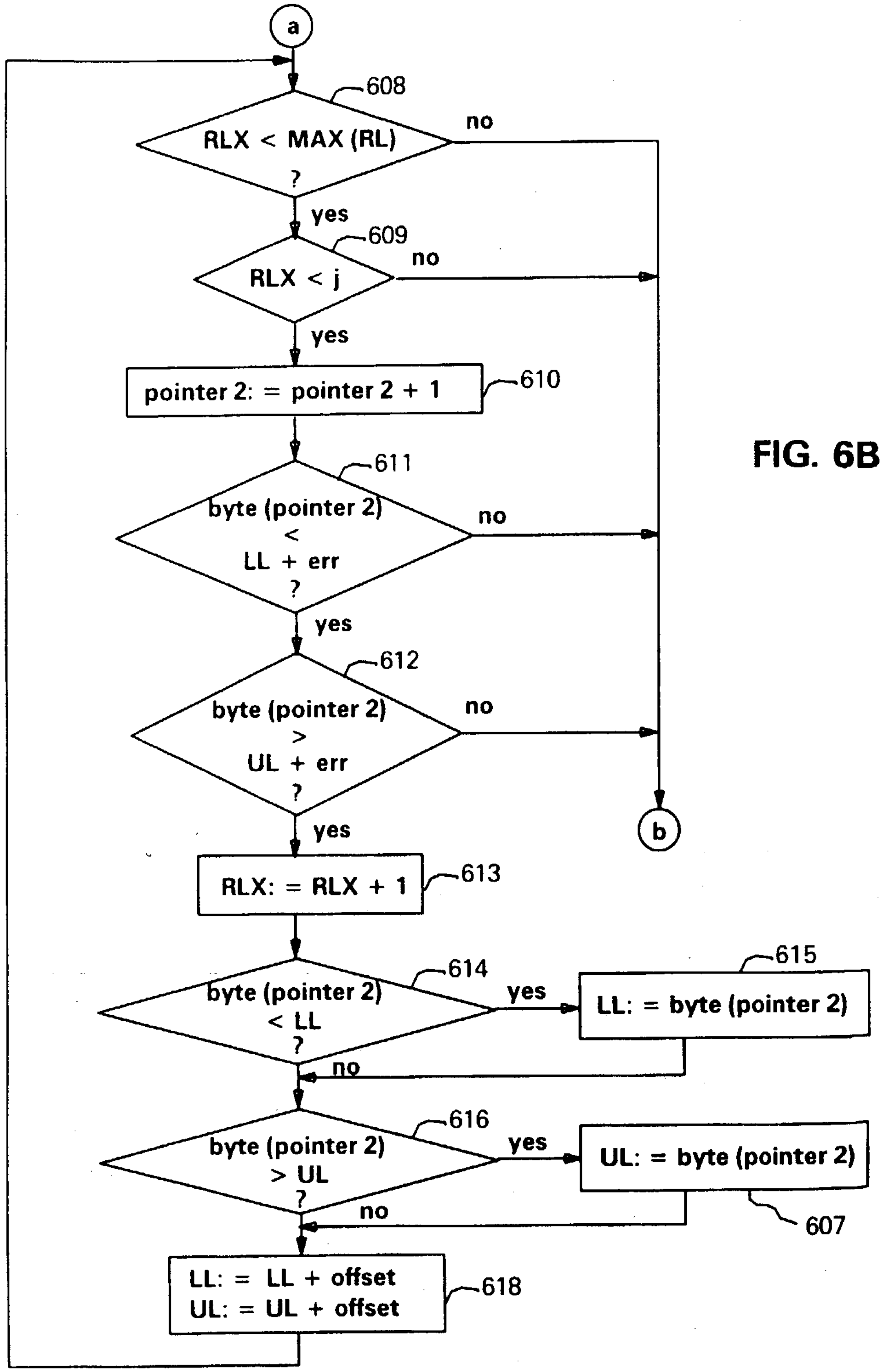
a
608
RLX < MAX (RL)
?
no
yes
609
RLX < j
no
yes
pointer 2: = pointer 2 + 1
610
611
byte (pointer 2)
<
LL + err
?
no
yes
612
byte (pointer 2)
>
UL + err
?
no
yes
b
RLX: = RLX + 1
613
614
byte (pointer 2)
< LL
?
yes
615
LL: = byte (pointer 2)
no
616
byte (pointer 2)
> UL
?
yes
UL: = byte (pointer 2)
607
no
LL: = LL + offset
UL: = UL + offset
618

FIG. 6B

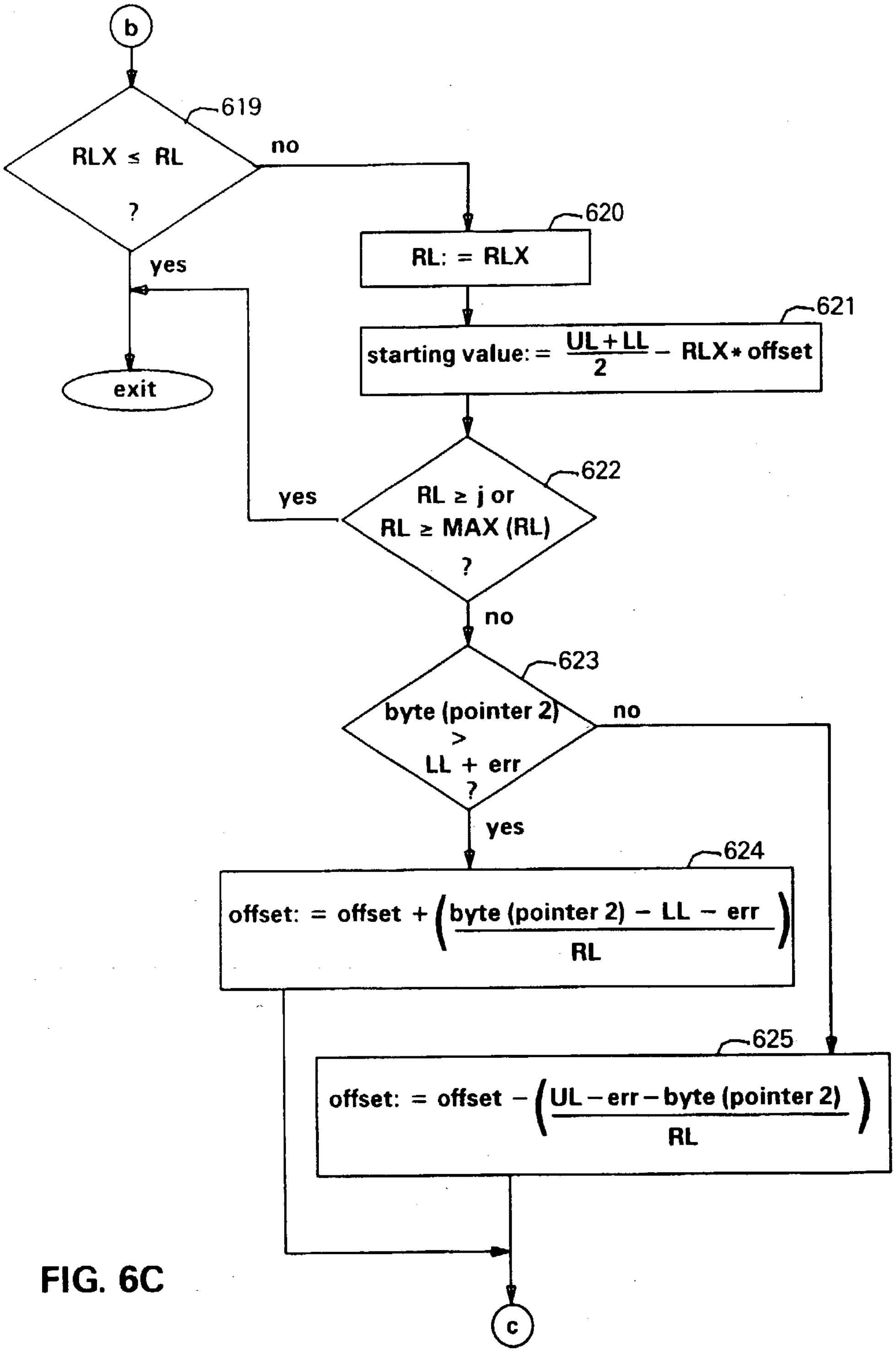
b
619
RLX ≤ RL ?
no
yes
620
RL: = RLX
621
starting value: = $\frac{UL+LL}{2}$ − RLX∗offset
exit
622
yes
RL ≥ j or RL ≥ MAX (RL) ?
no
623
byte (pointer 2) > LL + err ?
no
yes
624
offset: = offset + $\left(\frac{\text{byte (pointer 2)} - LL - err}{RL}\right)$
625
offset: = offset − $\left(\frac{UL - err - \text{byte (pointer 2)}}{RL}\right)$
c

FIG. 6C

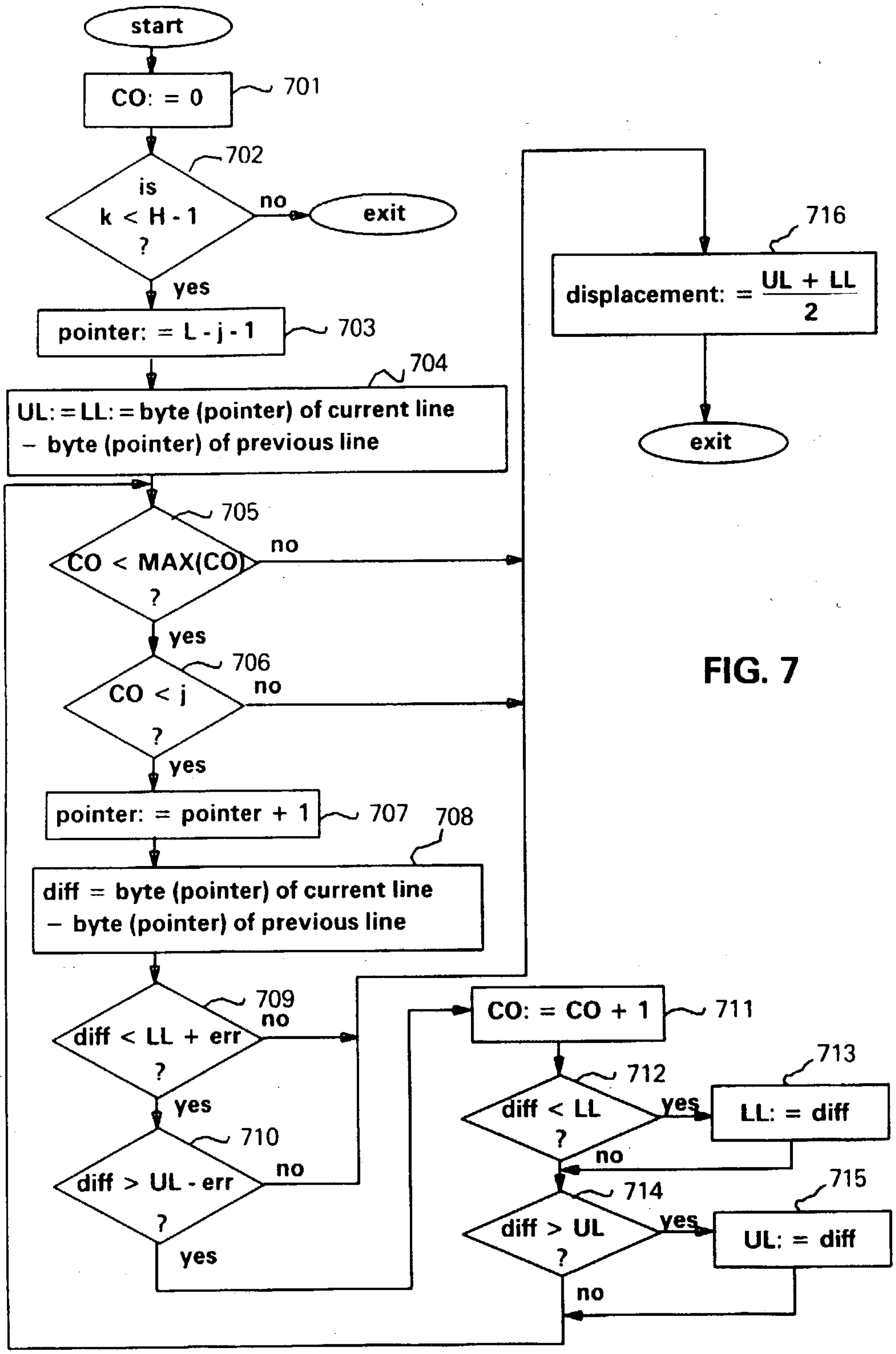
start
CO: = 0
701
702
is
k < H - 1
?
no
exit
yes
pointer: = L - j - 1
703
704
UL: = LL: = byte (pointer) of current line
– byte (pointer) of previous line
705
CO < MAX(CO)
?
no
yes
706
CO < j
?
no
yes
pointer: = pointer + 1
707
708
diff = byte (pointer) of current line
– byte (pointer) of previous line
709
diff < LL + err
?
no
yes
710
diff > UL - err
?
no
yes
CO: = CO + 1
711
712
diff < LL
?
yes
LL: = diff
713
no
714
diff > UL
?
yes
UL: = diff
715
no
716
displacement: = $\frac{UL + LL}{2}$
exit

FIG. 7

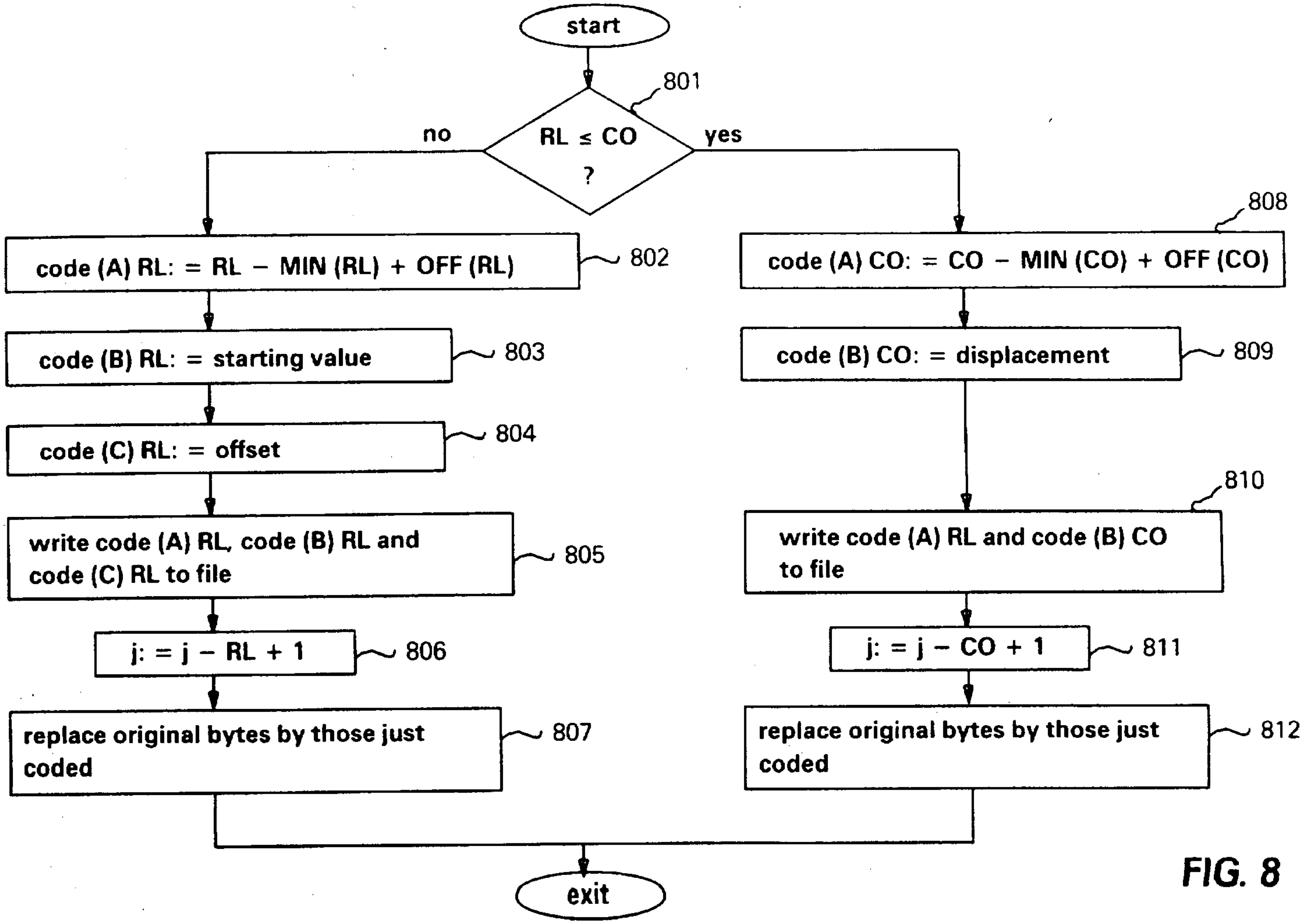
start
801
RL ≤ CO ?
no
yes
code (A) RL: = RL − MIN (RL) + OFF (RL)
802
code (B) RL: = starting value
803
code (C) RL: = offset
804
write code (A) RL, code (B) RL and code (C) RL to file
805
j: = j − RL + 1
806
replace original bytes by those just coded
807
code (A) CO: = CO − MIN (CO) + OFF (CO)
808
code (B) CO: = displacement
809
write code (A) RL and code (B) CO to file
810
j: = j − CO + 1
811
replace original bytes by those just coded
812
exit

FIG. 8

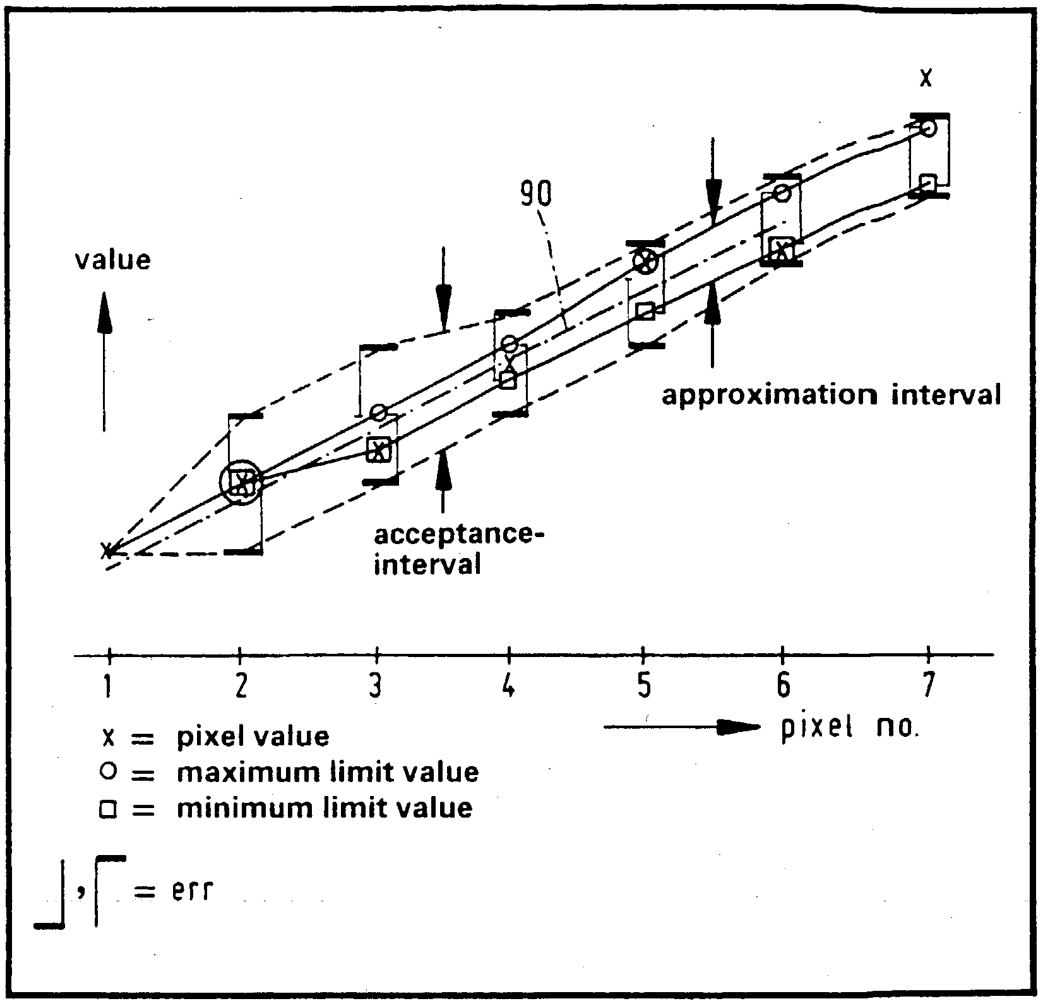
x
90
value
approximation interval
acceptance-
interval
1
2
3
4
5
6
7
pixel no.
x = pixel value
○ = maximum limit value
□ = minimum limit value
= err

FIG 9

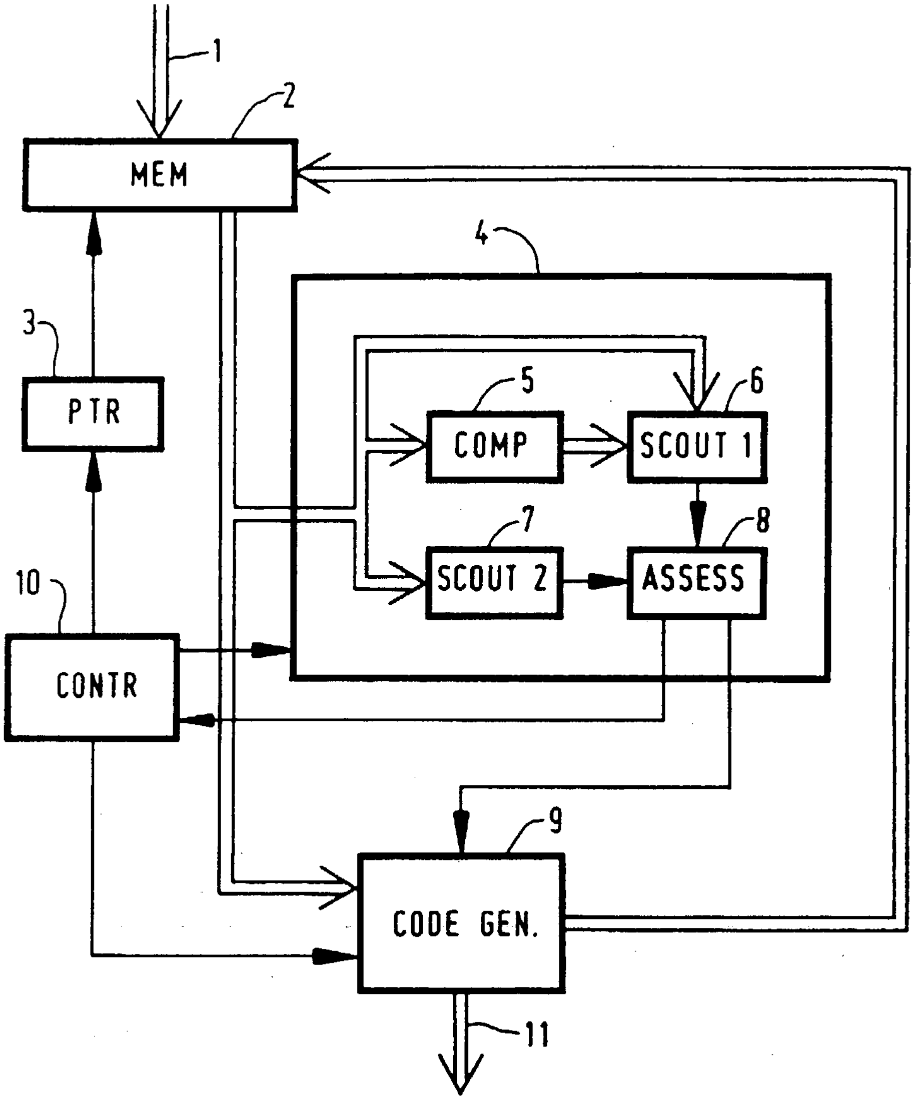
1
2
MEM
3
PTR
10
CONTR
4
5
COMP
6
SCOUT 1
7
SCOUT 2
8
ASSESS
9
CODE GEN.
11

FIG. 10

European Patent
Office

EUROPEAN SEARCH REPORT

Application Number

EP 92 20 0344

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
X	Supplement to IEEE Transactions on Aerospace and Electronic Systems, Vol. 2, No.4, July 1966, IEEE, USA, P.E. Drapkin: 'Video data compression', pages 392-400 * page 394, paragraph 3 - page 395 * * figures 6,188,19 * ---	1-10	H04N1/41 H04N7/137 H03M7/30
X A	WO-A-8 704 032 (BRITISH BROADCASTING CORPORATION) 2 July 1987 * abstract; claims 1-3,6 * * figures 1-3,6 * * page 8, line 7 - page 11, line 9 * ---	1-6 7-10	
A	Proceedings of the 1967 National Telemetering Conference, May 16-18, 1967, San Francisco, American Institute of Aeronautics and Astronautics, New York, USA, J.W. Stumpe: 73 'Redundancy reduction techniques and applications', pages 50-56 * page 51 - page 53 * ---	1-10	
A	IBM TECHNICAL DISCLOSURE BULLETIN vol. 30, no. 3, August 1987, NEW YORK, US pages 1246 - 1247; 'Reduction of data representing one pel line' * the whole document * ---	1-6,9,10	TECHNICAL FIELDS SEARCHED (Int. Cl.5) H04N H03M
A	WO-A-8 905 083 (TECHNOLOGY , INC.) 1 June 1989 * abstract; claims 1-7 * * page 1, line 1 - page 11, line 9 * * page 90, line 30 - page 92, line 8 * * figures 1,2,14 * -----	7,8	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	29 APRIL 1992	HOEKSTRA A.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document